Accelerating Deterministic Bit-Stream Computing with Resolution Splitting

M. Hassan Najafi[†,1], S. Rasoul Faraji[*,1], Bingzhe Li[*], David J. Lilja[*], and Kia Bazargan[*]
[†]School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA, USA
[*]Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, USA
najafi@louisiana.edu, faraj008@umn.edu, lixx1743@umn.edu, lilja@umn.edu, kia@umn.edu

***Abstract*—Deterministic approaches to stochastic computing (SC) have been recently proposed to produce completely accurate results with stochastic logic. Long processing time is the main limitation of these methods when a deterministic zero error rate output is expected. For instance, when multiplying two n-bit precision input values, a processing time of 2^{2n} cycles is required. This long processing time makes the current deterministic approaches of SC inefficient for many applications. In this work, we propose an acceleration method based on resolution splitting to mitigate this long latency. The result is an exponential reduction in the processing time at the cost of some increase in the hardware area. The exponential reduction in the processing time results in a significant reduction in energy consumption. Synthesis results show that for the common 2-input multiplier, the proposed design decreases the energy consumption more than 2000$\times$ compared to the prior state-of-the-art deterministic bit-stream-based design.**

***Keywords*-Stochastic Computing, deterministic bit-stream computing, resolution splitting, performance enhancement**

I. Introduction

Deterministic computation on stochastic bit-streams [7] [9] [10] has been recently introduced as an evolution of the stochastic computing (SC) paradigm [2] [6] [12]. SC has been known as a noise tolerant approximate computation approach. Recent work has shown that by properly structuring the inputs of the stochastic circuits, computation with stochastic constructs can be performed deterministically. The results are completely accurate with no random fluctuations or input correlations. Multiplication and scaled addition are the most common stochastic operations with simple logic implementations. An AND gate works as a multiplier and a multiplexer implements a scaled adder if their inputs are connected to independent bit-streams. The recently developed deterministic methods of computation on bit-streams provide this independence by relatively prime stream lengths [9], clock division, and rotation-based input generation methods [7]. The operations, however, must run for a large number of clock cycles to guarantee deterministic and completely accurate results; In case of operating on i n-bit resolution data represented by i 2^n-bit length independent bit-streams, the operation must run for 2^{in} cycles (i.e., the product of the length of bit-streams) to guarantee completely accurate result.

This large number of cycles required by deterministic approaches makes them energy inefficient for many applications [4]. Increasing the number of independent inputs further worsens the problem which means a very limited scalability. In this work, we propose a hybrid bitstream-binary resolution splitting technique to mitigate the long latency and so the high energy consumption problem of the deterministic computation on bit-streams. The result is an exponential decrease in the processing time and energy consumption at a reasonable increase in the hardware area cost. While the processing time and energy consumption are still higher than those of the conventional binary design counterparts, the numbers are comparable when the application can tolerate some level of inaccuracy and so allows more reduction in delay and energy. The noise-tolerance and the low hardware cost [12] advantages of bit-stream-based designs can further make this a wining proposition.

This paper is structured as follows: Section II presents background information on SC and the state-of-the-art deterministic bit-stream computing. In Section III, we describe our proposed method for performance enhancement of the deterministic bit-stream-based methods. In Section IV, we evaluate and compare the energy consumption of the proposed designs to conventional binary and also to previous bit-stream-based implementations. Finally, in Section V, we present conclusions.

II. Background

A. Stochastic Computing

Stochastic computing (SC) is a low-cost noise-tolerant computing paradigm [2], [6], [12]. Logical computation is performed on probability data in the [0, 1] interval. The data is represented by uniformly distributed (random) bit-streams. The ratio of the number of ones to the length of the bit-stream determines the value. For example, 1101011100 is a representation of 0.6 in the stochastic domain. A random, pseudo-random, or increasing number is compared to a constant number (based on the input data) and the output of comparison produces one bit of the stochastic bit-stream in each cycle [12]. Fig. 1 shows the structure of a stochastic bit-stream generator responsible for converting data from binary radix to bit-stream representation. The generated bit-streams are processed using stochastic logic and an output bit-stream is produced. The output bit-stream is converted back to the binary domain by simply counting the number of ones in the bit-stream using a binary counter.

Multiplication and scaled addition are two common stochastic operations with simple logic implementation and are widely used in image processing and neural network

[1]These authors contributed equally.

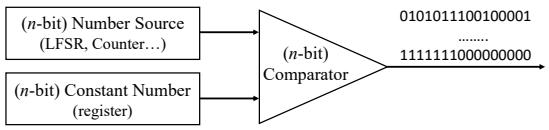

Fig. 1. Structure of a stochastic stream generator.

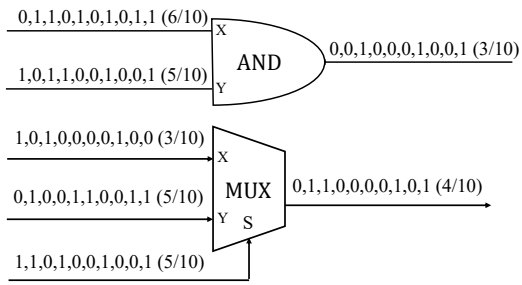

Fig. 2. Examples of stochastic multiplication and scaled addition.

applications. As shown in Fig. 2, an AND gate works as a stochastic multiplier and a multiplexer performs scaled addition if independent (uncorrelated) bit-streams are connected to their inputs. Due to the importance and wide application of these two basic stochastic operations we will introduce our hybrid resolution splitting method based on them.

B. Deterministic Bit-Stream Based Computing

The random fluctuation inherent in generating bit-streams and the correlation between bit-streams make the conventional SC an approximate computing paradigm [2][12]. Recently Jenson and Riedel in [7] and Najafi et al. in [9] and [10] showed that computation on stochastic bit-streams can be performed deterministically. By properly structuring input bit-streams, completely accurate results can be produced with no random fluctuation or correlation errors. In these deterministic approaches, the independence between bit-streams is provided by using relatively prime stream lengths [9], rotation, or clock division [7]. Low discrepancy (LD) sequences such as Sobol [8] and Halton [1] sequences can also be directly used to perform deterministic computation on bit-streams [11].

A common property to all of these state-of-the-art deterministic approaches is that, to produce correct results, the operations must run for an exact number of clock cycles (i.e., the product of the length of the input bit-streams) [7][10]. For example, to multiply two 8-bit precision input values represented using two 2^8-length bit-streams, the input bit-streams must be connected to the inputs of an AND gate and the operation must run for 2^{16} cycles. For the case of multiplying three and four 8-bit precision values, the operation time extends to 2^{24} and 2^{32} cycles, respectively. This long processing time makes the deterministic computation with bit-streams inefficient and unjustifiable for most applications.

III. Proposed Approach

In this work, we propose to enhance the performance of deterministic computation with bit-streams using resolution splitting of data. Instead of directly converting high resolution N-bit binary data into very long bit-streams, the data is split into K sub-values of N/K-bit resolution, where K is a power of 2 (i.e., 2, 4, ...). For example, for $N = 8$ and $K = 2$, input $x = 178$ with a binary representation of $X = 10110010$ is split into two sub-values of $X_0 = 1011$ and $X_1 = 0010$. For $K = 4$, this number is split into four sub-values of $X_0 = 10$, $X_1 = 11$, $X_2 = 00$, and $X_3 = 10$. For simplicity of presentation we define N/K as M. Resolution splitting of two input values X and Y with $K = 2$ gives [13]:

$$X = X_0 \cdot 2^M + X_1 \quad , \quad Y = Y_0 \cdot 2^M + Y_1$$

and with $K = 4$ gives:

$$X = X_0 \cdot 2^{3M} + X_1 \cdot 2^{2M} + X_2 \cdot 2^M + X_3$$
$$Y = Y_0 \cdot 2^{3M} + Y_1 \cdot 2^{2M} + Y_2 \cdot 2^M + Y_3$$

Each one of these M-bit sub-values is converted to a bit-stream representation using a binary to bit-stream converter (Fig. 1). Deterministic computation is performed on the generated bit-streams and the output bit-streams are converted back to binary format and accumulated to produce the final value. In what follows we discuss our proposed resolution splitting method for the two main stochastic operations, multiplication and scaled addition.

A. Multiplication

$$Z = X \cdot Y$$

For multiplication operation we exploit the idea of performing a full precision multiplication by first producing partial products. This will require simpler partial multipliers. Since partial products have a lower resolution, they are represented by exponentially shorter bit-streams. Consequently, producing them using deterministic bit-stream computation requires an exponentially smaller number of cycles. Partial products with bit-stream representation are then converted to binary radix format and added together to produce the final result. For resolution splitting with $K = 2$ we have:

$$Z = (X_0 \cdot Y_0) \cdot 2^{2M} + (X_0 \cdot Y_1 + X_1 \cdot Y_0) \cdot 2^M + X_1 \cdot Y_1$$

Figs. 3, 4, and 5 show the conventional circuit for deterministic bit-stream multiplication ($K = 1$), the proposed circuit for resolution splitting with $K = 2$, and the general proposed circuit for resolution splitting with any K, respectively. For each design, two random number generators are necessary to convert the input data to bit-stream representation. To evaluate the efficiency of the proposed hybrid designs we synthesize these circuits as well as the conventional binary implementation for the case of 2-input, 3-input, and 4-input 8-bit precision multiplication. We use the Synopsys Design Compiler vH2013.12 with a 45nm gate library to synthesize each design. The synthesis results are reported in Table I.

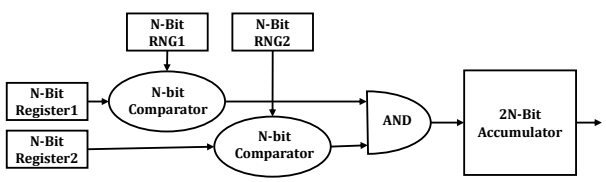

Fig. 3. Conventional circuit for bit-stream-based multiplication (K=1)

TABLE I
SYNTHESIS RESULTS OF DIFFERENT DESIGN APPROACHES FOR MULTIPLICATION OF 8-BIT PRECISION INPUT DATA

Inputs	Design Approach	Area (μm^2)	Delay (ns)	Power(@Maxf) mW	Energy/cycle(@Maxf)	Operation Cycles	Total Energy(pJ)
2	Conv. Binary	1021	0.80	2.040	1.632	1	1.6
	Bit-stream K=1	359	0.38	1.055	0.400	2^{16}	26,273.3
	Bit-stream K=2	367	0.48	1.012	0.485	2^{8}	124.3
	Bit-stream K=4	483	0.57	1.315	0.749	2^{4}	11.9
3	Conv. Binary	2,909	1.30	3.155	4.101	1	4.1
	Bit-stream K=1	541	0.42	1.386	0.582	2^{24}	9,769,372.8
	Bit-stream K=2	569	0.58	1.169	0.678	2^{12}	2,777.0
	Bit-stream K=4	1,351	0.88	1.561	1.373	2^{6}	87.9
4	Conv. Binary	5,603	1.55	6.002	9.303	1	9.3
	Bit-stream K=1	727	0.43	1.786	0.768	2^{32}	3,299,823,374.0
	Bit-stream K=2	872	0.66	1.364	0.900	2^{16}	59,028.2
	Bit-stream K=4	3,895	1.22	2.184	2.664	2^{8}	682.0

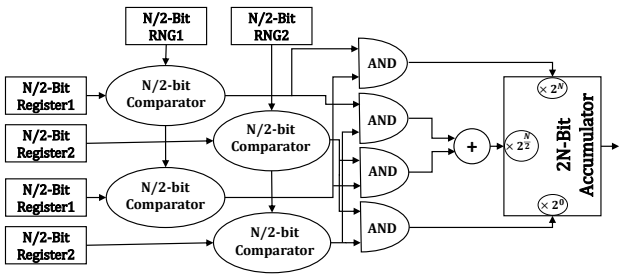

Fig. 4. Proposed circuit for bit-stream-based multiplication with resolution splitting of $K = 2$

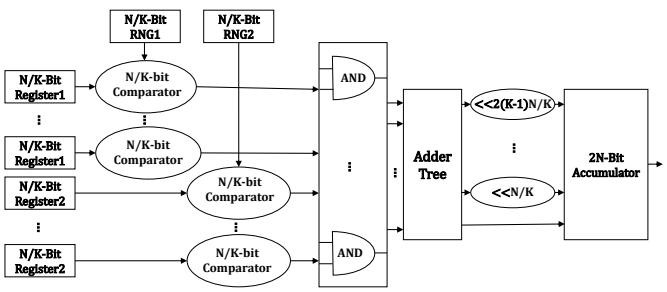

Fig. 5. General proposed circuit for bit-stream based multiplication with resolution splitting of K

We do not consider the cost of the random number generators since the cost depends on the selected deterministic approach. For example, for the deterministic approaches of processing unary bit-streams proposed in [7], two binary counters and for the deterministic approaches of processing pseudo-random bit-streams proposed in [10], two linear feedback shift registers are required. For many applications of SC such as neural network and image processing applications, these number generators will be shared between a large number of multiplier units and so their effective cost will be insignificant.

As can be seen in the synthesis results, all bit-stream-based implementations have a lower hardware area cost than their corresponding conventional binary implementation. This lower cost is mainly due to low-cost implementation of multiplication operation using simple AND gates. The hardware cost, however, increases by increasing the number of inputs, the precision of the input data, and K. A similar trend is also observed for the delay (i.e., critical path latency), the power consumption, and the energy consumption per cycle.

The main weakness of the bit-stream-based implementations is its high processing time (delay×number of cycles) and high energy consumption (energy per cycle×number of cycles). Comparing the total energy consumption of the conventional binary and the previous deterministic bit-stream implementation ($K = 1$) we see more than 16×10^3, 23×10^5, and 35×10^7 times increase when implementing 2-input, 3-input, and 4-input multipliers, respectively. This high energy consumption is unacceptable for any application. With the proposed resolution splitting method, however, the energy consumption significantly decreases due to an exponential decrease in the number of operation cycles.

For 2-input multiplication, the processing time decreases from 2^{16} cycles for the previous method ($K = 1$) to 2^8 cycles for the proposed method with $K = 2$ and to 2^4 cycles for the proposed method with $K = 4$. For the case of $K = 4$, the increase rates of the energy consumption decrease from 16×10^3 to 12 for 2-input, from 23×10^5 to 21 for 3-input, and from 35×10^7 to 92 for 4-input multiplication. Although by increasing the number of inputs the gains from the proposed method decreases, still the 2-input multiplier is widely used in different applications of the bit-stream-based designs [5]. With the savings from the proposed method, the total processing time and the total energy consumption become acceptable for many applications. When the application itself can tolerate small rates of inaccuracy, the processing time and hence energy consumption can be further reduced. Factoring in the noise-tolerance and the low hardware cost advantages of the bit-stream-based design, the proposed designs become an attractive alternative to the conventional binary design.

B. Scaled Addition

$$Z = (1 - S) \cdot X \ + \ S \cdot Y$$

In the bit-stream domain, numbers are limited to the [0,1] interval. This makes the normal add operation inconvenient because the sum of two numbers will lie in the [0,2] interval. Scaled addition is therefore used to map the results back to [0,1]. For scaled addition of two inputs a 2-input multiplexer with a scaling factor of $S = 1/2$ is used. This changes the effective operation to $Z = (X + Y)/2$. For scaled addition of four (eight) inputs, a 4-input (8-input) multiplexer with two (three) scaling inputs of $1/2$ are used. This changes the effective operations to $(X + Y + A + B)/4$ for the 4-input

TABLE II
SYNTHESIS RESULTS OF DIFFERENT DESIGN APPROACHES FOR SCALED ADDITION OF 8-BIT PRECISION INPUT DATA

Inputs	Design Approach	Area (μm^2)	CP (ns)	Power(@Maxf) mW	Energy/cycle(@Maxf)	Operation Cycles	Total Energy(pJ)
2	Conv. Binary	159	0.40	0.851	0.341	1	0.3
	Bit-stream K=1	174	0.36	0.652	0.235	2^9	120.2
	Bit-stream K=2	191	0.34	0.765	0.260	2^5	8.3
	Bit-stream K=4	210	0.34	0.911	0.310	2^3	2.5
4	Conv. Binary	314	0.60	0.958	0.575	1	0.6
	Bit-stream K=1	187	0.39	0.607	0.237	2^{10}	242.4
	Bit-stream K=2	201	0.36	0.776	0.279	2^6	17.9
	Bit-stream K=4	230	0.38	0.829	0.315	2^4	5.0
8	Conv. Binary	611	0.74	1.616	1.195	1	1.2
	Bit-stream K=1	210	0.43	0.563	0.242	2^{11}	496.1
	Bit-stream K=2	237	0.43	0.729	0.314	2^7	40.1
	Bit-stream K=4	324	0.43	0.779	0.335	2^5	10.7

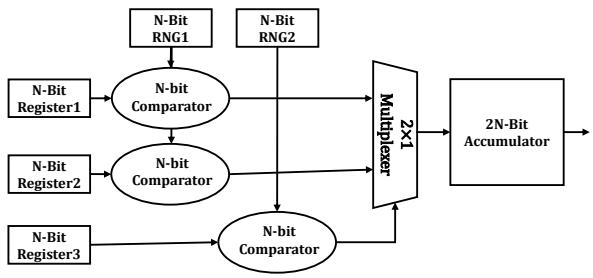

Fig. 6. Conventional circuit for bit-stream-based scaled addition (K=1)

and to $(X+Y+A+B+C+D+E+F)/8$, for the 8-input case.

Similar to the multiplication operation, for the scaled addition we split the full precision operation to simpler operations with a lower resolution. Partial additions in a lower resolution are performed on exponentially shorter bit-streams. The deterministic output bit-streams are therefore ready in exponentially smaller number of cycles. The difference with the multiplication operation is that, here the produced output bit-streams are converted to binary radix representation and concatenated to produce the final result. For resolution splitting of 2-input scaled addition with $K = 2$ we have [13]:

$$Z = (1-S)\cdot(X_0\cdot 2^M + X_1) + S\cdot(Y_0\cdot 2^M + Y_1) = ((1-S)\cdot X_0 + S\cdot Y_0)\cdot 2^M + ((1-S)\cdot X_1 + S\cdot Y_1) = Z_0\cdot 2^M + Z_1$$

Figs. 6, 7, and 8 show the previous circuit for deterministic bit-stream-based scaled addition ($K = 1$), the proposed circuit for resolution splitting with $K = 2$, and the general proposed circuit for resolution splitting with any K, respectively. To evaluate the efficiency of the proposed designs we synthesized these circuits and also the conventional binary implementation of the scaled addition operation for the cases of adding two, four, and eight 8-bit precision input data. The synthesis results are reported in Table II.

An important property of the bit-stream-based scaled addition is a loss of precision if for N-bit precision input data (2^N-bit input streams) the same bit precision output (2^N-bit output stream) is produced. In such cases parts of the information in the input bit-streams are effectively discarded [3]. As an example, consider the case of performing scaled addition on

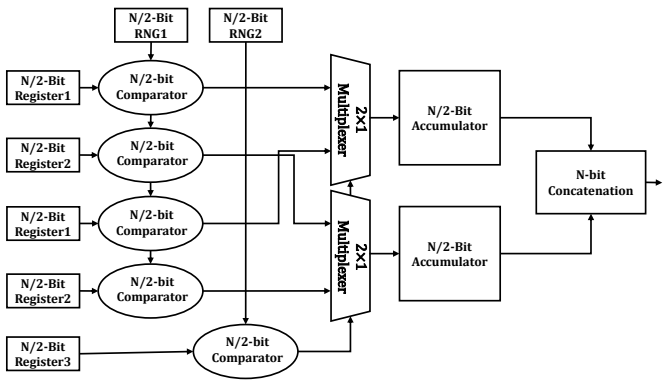

Fig. 7. Proposed circuit for bit-stream based scaled addition with resolution splitting of $K = 2$

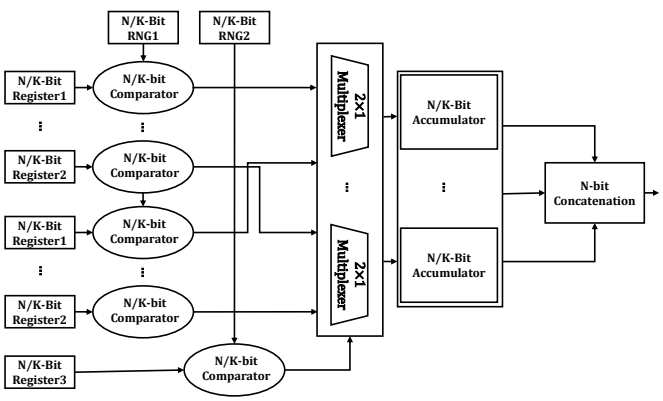

Fig. 8. General proposed circuit for bit-stream based scaled addition with resolution splitting of K

$3/256$ and $4/256$, each represented by a 2^8-bit length bit-stream. The expected output value is $7/512$. Producing 2^8-bit output bit-stream results in a loss of precision because $7/512$ cannot be precisely presented using a 2^8-bit stream. A 2^9-bit stream is necessary to precisely present this output value. So for bit-stream-based scaled addition of two 8-bit precision data, a processing time of 2^9 cycles for $K = 1$, 2^5 cycles for $K = 2$, and 2^3 cycles for $K = 4$ is required. The total number of cycles required for producing accurate results by each one of the implemented bit-stream-based scaled addition circuits is reported in Table II.

As reported in Table II, the 4- and 8-input bit-stream-based designs have a lower hardware area cost, critical path,

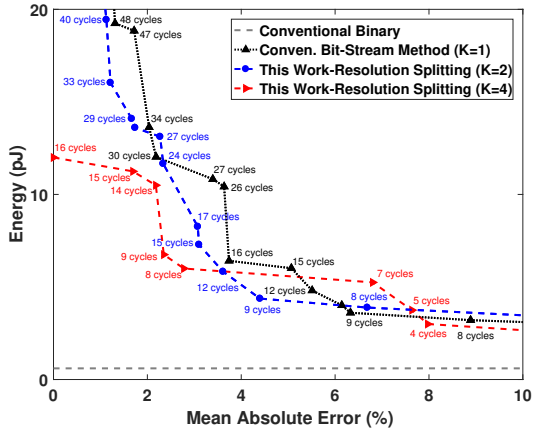

Fig. 9. Energy consumption vs. MAE of the implemented 2-input 8-bit precision multipliers

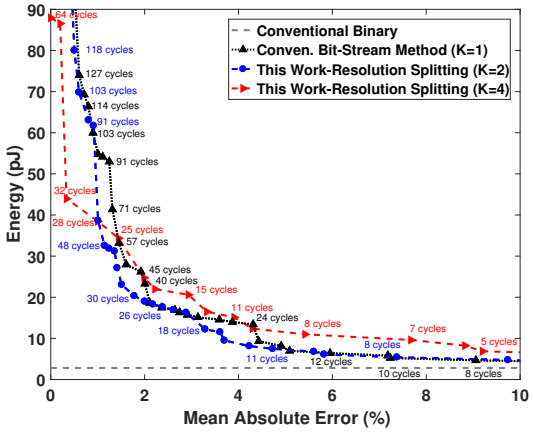

Fig. 10. Energy consumption vs. MAE of the implemented 3-input 8-bit precision multipliers

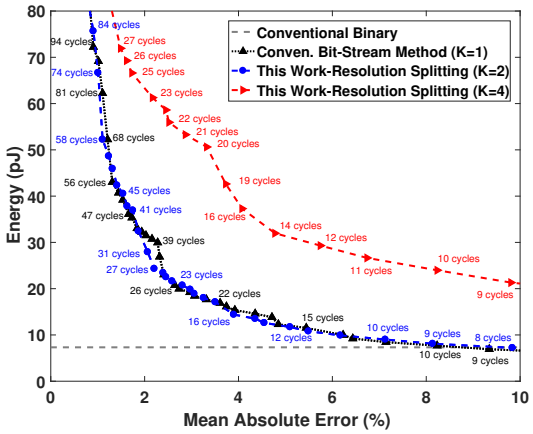

Fig. 11. Energy consumption vs. MAE of the implemented 4-input 8-bit precision multipliers

and power consumption than their conventional binary counterparts. Due to an exponential decrease in the processing time, for the case of producing completely accurate results, the proposed designs with $K = 2$ and $K = 4$ consume a significantly lower total energy compared to the previous bit-stream-based design ($K = 1$). For example, for the 8-input scaled addition, the proposed designs with $K = 2$ and $K = 4$ result in more than $12\times$ and $46\times$ reduction in the total energy consumption, respectively.

Although the processing time and the energy consumption are still higher than the conventional binary counterpart, the proposed designs provide a comparable or lower hardware cost while are also inherently more fault tolerant than the binary design. In the next section, we show that if the application can tolerate some level of inaccuracy, the total energy consumption of the proposed designs can become comparable to and in some cases even lower than that of the binary design.

IV. Experimental Results

In this section, we evaluate the energy efficiency of the proposed designs for applications that can tolerate some rates of inaccuracy. We evaluate and compare the energy consumption of the 8-bit precision 2-, 3-, and 4-input multipliers and also 2-, 4-, and 8-input scaled adders when implemented with the conventional binary design, the previous deterministic bit-stream-based designs without resolution splitting ($K = 1$) and the proposed designs with resolution splitting of $K = 2$ and $K = 4$. We find the energy consumption of each design in achieving different mean absolute error (MAE) rates.

For the two random number generators required in the proposed designs we use the first two low-discrepancy (LD) Sobol sequences from the MATLAB built-in Sobol sequence generator. The random numbers from these number generators will be compared to the input data to convert them from binary to bit-stream representation. With LD sequences, 1s and 0s in the bit-streams are uniformly spaced, so the streams do not suffer from random fluctuations.

The number of cycles required by each design to produce completely accurate results (MAE of 0%) was reported in Tables I and II. Running the operations for a lower number of cycles than the required one results in truncation errors in the computation and hence an increase in the MAE. We extracted the number of cycles required by each bit-stream-based design to achieve each level of MAE by exhaustively testing the performance of each circuit by a large number of random input values. We multiplied the extracted number of cycles by the energy per cycle values reported in Tables I and II to find the total energy consumptions of each design.

Figs. 9-14 compare the total energy consumption of each design to achieve different accuracy levels. As can be seen, in general, the proposed technique with $K = 2$ resulted in a better or at least the same energy level as the previous bit-stream based design ($K = 1$). The resolution splitting method with $K = 4$ in most cases could similarly result in energy saving compared to the conventional case with $K = 1$. The only exception is the 4-input multiplier that the proposed design with $K = 4$ consumes a higher energy consumption for MAEs of greater than 1%. The important point, however, is that for the cases that the proposed designs provide the same level or higher energy consumption than the previous bit-stream-based design ($K = 1$), a significantly lower processing time is achieved with the resolution splitting method.

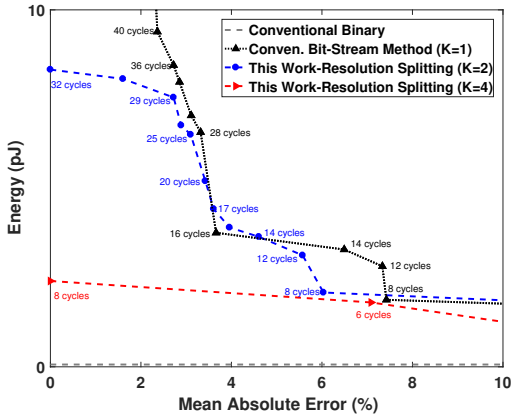

Fig. 12. Energy consumption vs. MAE of the implemented 2-input 8-bit precision scaled adders

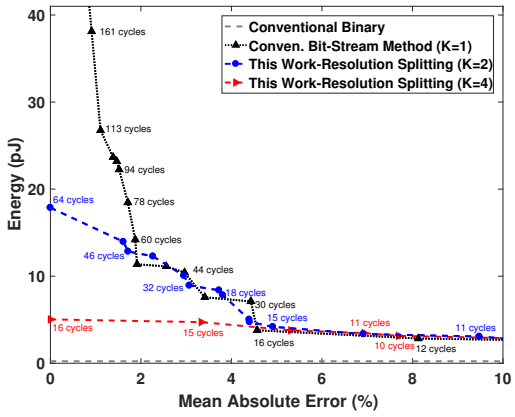

Fig. 13. Energy consumption vs. MAE of the implemented 4-input 8-bit precision scaled adders

We added a dashed line to each figure to compare different bit-stream based designs with the binary implementation. As can be seen, when the application can tolerate some level of inaccuracy, the energy consumption of the proposed designs can be reduced to some levels very close to the energy consumption of the conventional binary design. Although a higher energy consumption compared to the conventional binary designs is still a drawback for the deterministic bit-stream based designs, the proposed design method could mitigate this drawback by significantly decreasing the processing time and also improving the energy consumption in most cases. Reducing the hardware cost, particularly for implementing the multiplication operation, and also the inherent noise tolerance of computations on bit-streams, are the advantages of the proposed designs to the conventional binary implementation.

V. Conclusion

This work proposes a hybrid bit-stream-binary approach to improve the long processing time and the high energy consumption of the recently developed deterministic methods of computation with bit-streams. By resolution splitting of input data an exponential reduction in the processing time and energy consumption is achieved. We developed hardware architectures for two important operations of bit-stream-based computing, multiplication and scaled addition. Low cost, high performance, and yet energy-efficient implementation of these

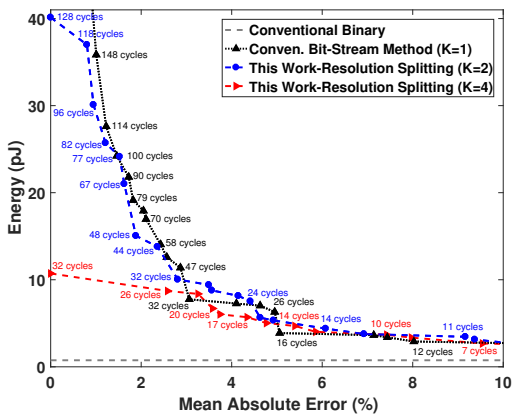

Fig. 14. Energy consumption vs. MAE of the implemented 8-input 8-bit precision scaled adders

operations are most useful for applications that can tolerate slight inaccuracy such as neural network and image processing applications.

Acknowledgment

This work was supported in part by National Science Foundation grant no. CCF-1438286. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] A. Alaghi and J. Hayes. Fast and accurate computation using stochastic circuits. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–4, March 2014.

[2] A. Alaghi and J. P. Hayes. Survey of stochastic computing. *ACM Trans. Embed. Comput. Syst.*, 12(2s):92:1–92:19, 2013.

[3] A. Alaghi, W. Qian, and J. P. Hayes. The Promise and Challenge of Stochastic Computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(8):1515–1531, Aug 2018.

[4] S. R. Faraji and K. Bazargan. Hybrid binary-unary hardware accelerators. In *2019 24th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2019.

[5] S. R. Faraji, M. H. Najafi, B. Li, K. Bazargan, and D. J. Lilja. Energy-Efficient Convolutional Neural Networks with Deterministic Bit-Stream Processing. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2019*, pages 1–6, March 2019.

[6] B. Gaines. Stochastic computing systems. In *Advances in Information Systems Science*, pages 37–172. Springer US, 1969.

[7] D. Jenson and M. Riedel. A Deterministic Approach to Stochastic Computation. In *Proceedings of the 35th International Conference on Computer-Aided Design*, ICCAD '16, pages 102:1–102:8, 2016.

[8] S. Liu and J. Han. Energy efficient stochastic computing with sobol sequences. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 650–653, March 2017.

[9] M. H. Najafi and S. Jamali-Zavareh and D. J. Lilja and M. D. Riedel and K. Bazargan and R. Harjani. Time-Encoded Values for Highly Efficient Stochastic Circuits. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 25(5):1–14, 2017.

[10] M. H. Najafi and D. Lilja. High Quality Down-Sampling for Deterministic Approaches to Stochastic Computing. *IEEE Transactions on Emerging Topics in Computing*, 2018.

[11] M. H. Najafi, D. J. Lilja, and M. Riedel. Deterministic Methods for Stochastic Computing using Low-Discrepancy Sequences. In *Proceedings of the 37th International Conference on Computer-Aided Design*, ICCAD '18, pages 1–8, 2018.

[12] W. Qian, X. Li, M. Riedel, K. Bazargan, and D. Lilja. An architecture for fault-tolerant computation with stochastic logic. *Computers, IEEE Trans. on*, 60(1):93–105, Jan 2011.

[13] Y. Zhu, P. Suo, and K. Bazargan. Binary stochastic implementation of digital logic. In *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, FPGA '14, pages 171–180, New York, NY, USA, 2014. ACM.